OIPE JC88 JAN 1 8 2005 PATENT & TRADEMARK OFFICE

[40101/01801 - 2000.049 CP1]

AF
IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s)	:	David REYNA et al.
Serial No.	:	09/845,414
Filing Date	:	April 30, 2001
For	:	SYSTEM AND METHOD FOR UTILIZATION OF A COMMAND STRUCTURE REPRESENTATION
Group Art Unit:	:	2124
Examiner	:	John Q. Chavis

Mail Stop: Appeal Brief-Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Mailing
I hereby certify that this correspondence is being deposited with U.S. Postal Services as first class mail in an envelope addressed to:
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
By: [illegible] Date: January 12, 2005
Michael J. Marcin, (Reg. No. 48,198)

TRANSMITTAL

In response to the Notice of Appeal filed on October 13, 2004, please find an Appeal Brief for filing in the above-identified application.

The Commissioner is hereby authorized to charge the one-month extension fee in the amount of **$120.00** and any additional fees to the Deposit Account of **Fay Kaplun & Marcin, LLP No.** 50-1492. A copy of this paper is enclosed for that purpose.

Respectfully submitted,

[illegible]

Dated: January 12, 2005

Michael J. Marcin, Reg. No. 48,198

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, New York 10038

01/19/2005 SSESHE1 00000057 501492 09845414
01 FC:1251 120.00 DA

PATENT
Attorney Docket No.: 40101-01801

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	)	
	)	
David REYNA et al.	)	
	)	
Serial No.: 09/845,414	)	Group Art Unit: 2124
	)	
Filed: April 30, 2001	)	Examiner: John Q. Chavis
	)	
For: SYSTEM AND METHOD FOR UTILIZATION OF A COMMAND STRUCTURE REPRESENTATION	)))	**Board of Patent Appeals and Interferences**

Commissioner for Patents
P.O. Box 1450
Arlington, VA 22313-1450

Sir:

APPEAL BRIEF UNDER 37 C.F.R. § 1.192

In support of the Notice of Appeal filed October 13, 2004, and pursuant to 37 C.F.R. § 1.192, appellants present in triplicate their appeal brief in the above-captioned application.

This is an appeal to the Board of Patent Appeals and Interferences from the Examiner's final rejection of claims 1-19 in the final Office Action dated June 15, 2004. The appealed claims are set forth in the attached Appendix A.

1. Real Party in Interest

This application is assigned to Wind River Systems, Inc., the real party in interest.

2. Related Appeals and Interferences

There are no other appeals or interferences which would directly affect, be directly affected, or have a bearing on the instant appeal.

3. Status of the Claims

Claims 1-19 have been rejected in the final Office Action.

Appellants submitted an Amendment After Final on October 13, 2004 which cancelled claims 10 and 19, and amended claims 1, 2, 7, 8, 11, and 12. The Examiner issued an Advisory Action on December 15, 2004, which stated that the proposed amendments would be entered for purposes of Appeal. Thus, claims 1-9 and 11-18 as presented in the October 13, 2004 Amendment After Final are involved in this appeal. The appealed claims are included as Appendix A to this Appeal Brief.

4. Status of Amendments

All amendments submitted by the appellants have been entered.

5. Summary of the Invention

The present invention relates to representation of data file/codes using a command structure. (See Specification, page 2, ¶ 0005). Use of such command data structure representation is beneficial in that it allows a number of high level actions or operations to be performed upon the data file/codes. (See Specification, page 25, ¶ 0042). This invention may be used with any operating system that supports a Command Line Interface ("CLI"). (See

Specification, page 4, ¶ 0007).

The invention comprises a method for generating a list of desired elements of a first software code; extracting the desired elements from the first code; and performing an operation on the extracted elements. (See Specification, page 2, ¶ 0005). The first code used in this method would have a predefined structure. (See Specification, page 2, ¶ 0005). The invention corresponds to a system comprising three engines: a first engine which receives a list of desired elements of a first software code; a second engine which extracts the desired elements from the first code; and a third engine which performs an operation on the extracted elements. (See Specification, page 2, ¶ 0005).

An illustrative example of how the command data structure representation may be utilized according to the present invention is provided in Fig. 15. In the initial step, 710, a command data structure representation (i.e. 150; 200; 280), which includes command nodes, is generated. In the subsequent step, 712, an element is extracted from the command node. This extracted element must be found on a list of predetermined elements in order to proceed to step 716. If the element extracted in step 712 is not on the predetermined list, the user may extract another element from the command node, provided that there are any remaining. However, this second extracted element similarly must be found on the predetermined list of elements in order to proceed to step 716, which is the step of performing a set of predefined actions. The user may then extract another element from the command node and repeat the process, provided that there is an element remaining in the command node.

The command nodes used in the command data structure representation may be stored in a database, which may also include information about a particular command node. (See Specification, page 6, ¶ 0009). Additionally, command nodes may be edited, inserted, or deleted at any level of the structure. (See Specification, page 6, ¶ 0010). A developer may visually manipulate the command structure by adding or deleting command nodes. (See Specification, page 6, ¶ 0010).

Command nodes may also have associated handler functions and parameters, which can be edited. (See Specification, page 8, ¶ 0012). For example, Figure 4 shows "command3" node 240 as having two associated parameters: "unnamed1" parameter 241 and "unnamed2" parameter 242. Also, "command3" node 240 is seen to have an associated handler function, "VD_Command3Handler" 243. However, a command node is not required to have a handler function or a parameter. (See Specification, ¶¶ 0017, 0020).

An exemplary process for adding a command node to a command tree is shown in Figure 6. The developer first selects the location for the new command node. He then provides information about the new command node by inserting parameters and handler functions. In step 350, the command structure is generated using information entered by the developer. (See Specification, page 14, ¶ 0024). A file containing the information for the handler functions and parameters is then generated using input from the developer. (See Specification, page 16, ¶ 0026). Lastly, the handler function code is generated using information from the developer and the definitions generated in the preceding step. (See Specification, page 18, ¶ 0028). Upon completion of these steps, a new command node will have been inserted.

The structure of the command nodes must be defined prior to operation of the present invention, as this predefined command structure is a significant attribute of the first software code. The predefined command structure is also to be displayed via a graphical user interface.

6. Issues

I. Whether claims 1, 6, and 9 are unpatentable under 35 U.S.C. § 102(b) as being anticipated by the public use of a person sitting at his desk and adding two numbers that were selected from a list of numbers.

II. Whether claims 1, 6, and 9 are unpatentable under 35 U.S.C. § 102(b) as being anticipated by the public use of a person generating a playlist of songs on his computer, extracting one or more of the songs from the list, and saving or deleting the songs.

III. Whether claims 1-9 and 11-18 are unpatentable under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,966,536 to Ravichandran (the "Ravichandran reference").

7. Grouping of Claims

Claims 1-9 and 11-18 may stand or fall together.

8. Argument

I. The Rejection of Claims 1, 6, and 9 Under 35 U.S.C. § 102(b) as Being Anticipated by the Public Use of a Person Sitting at His Desk and Adding Two Numbers Should Be Reversed.

The Examiner rejected claims 1, 6, and 9 in the Final Office Action as being anticipated by the public use of a person sitting at a desk and adding two numbers. (See 6/15/04 Office Action, ¶ 1). This public use was further described in the Examiner's First Office Action, where he explained that the numbers used to perform the mathematical operation were selected from a list of numbers. (See 12/22/03 Office Action, ¶ 1).

Appellants have previously amended claim 1 to clarify the inventive method. The clarification highlights that the inventive method is executed on a computing device. The clarification further denotes that the elements are extracted from software code.

As amended, the inventive method could not have been anticipated by a person sitting at his desk and adding two numbers. The limitation of "A method *executed on a computing device*" applies to each step. (Claim 1). Furthermore, Appellants completely fail to understand how a person sitting at their desk and adding two numbers discloses a "first code having a predefined command structure" as recited in claim 1. As the present invention executes each step on a computing device and specifically recites software code having a defined structure, appellants respectfully submit that it is not anticipated by the public use of a person sitting at his desk and adding two numbers.

II. The Rejection of Claims 1, 6, and 9 Under 35 U.S.C. § 102(b) as Being Anticipated by the Public Use of a Person Generating a Playlist of Songs on His Computer, Extracting One or More of the Songs from the List, and Saving or Deleting the Songs Should be Reversed.

The Examiner rejected claims 1, 6, and 9 in the Final Office Action as being anticipated by the public use of a person generating a playlist of songs on his computer, extracting one or more of the songs from the list, and saving/deleting the songs to enable replays of only the remaining songs on the playlist. (6/15/04 Office Action, ¶ 1).

In his rejection, the Examiner appears to be equating an original playlist with a first software code. One would usually not consider a playlist and a software code to be equivalent. Software code is defined as a set of written computer instructions, as it is also known to those of ordinary skill in the art. A playlist, however, is not a set of written computer instructions. A playlist is defined as a list of musical tracks, which as the examiner points out may be saved or deleted. (6/15/04 Office Action, ¶ 1).

Claim 1 initially recites "generating a list of desired elements of the first software code." The specification of the present invention provides a description of the types of "elements" which are included in the software code (e.g., header elements, parameter elements, handler elements, etc.). (See Specification, page 26, ¶ 0044). The types of elements described and claimed in the present invention are completely different from a list of songs or an individual song on a list of songs.

Claim 1 further recites "the first code having a predefined *command structure*."

Appellant also gives several examples of a command structure in the specification. One such example is an XML language format. (Specification, page 26, ¶ 0043). To follow through with the Examiner's logic, the entire list of songs would be equated with the software code. But a list of songs does not have structure, it is a listing. The Examiner states that the name, number, location, type, and length of a playlist provide a defined structure. (Advisory Action, page 2). This is not a structure as claimed in the present invention. It is merely a display device used to show characteristics of a particular file.

In light of the aforementioned distinctions between an original playlist and a first software code, it is respectfully submitted that the present invention is not anticipated by the public use of a user generating a list of songs, selecting a song, and saving or deleting the song.

III. The Rejection of Claims 1-9 and 11-18 Under 35 U.S.C. § 102(b) as Being Anticipated by U.S. Patent No. 5,966,536 to Ravichandran Should be Reversed.

The Examiner rejected claims 1-19 in the Final Office Action as being anticipated by U.S. Patent No. 5,966,536 to Ravichandran (the "Ravichandran reference"). (6/15/04 Office Action, ¶ 2). The Ravichandran reference relates to a method and apparatus for transforming source executable code into target executable code. (Ravichandran, Abstract). In this process, the source executable is converted into a functionally equivalent source executable, and execution performance information is collected for each basic block of code therein and in the target executable. (Ravichandran, Abstract). An optimization metric is then performed on each basic block of code in the functionally equivalent source executable code and in the target executable code, and these optimization metrics are used to compare the basic blocks of the two codes. (Ravichandran, Abstract). Upon determination of which basic blocks have higher performance (the functionally equivalent source executable or the target executable), a target executable code is generated using a combination thereof. (Ravichandran, Abstract).

Claim 1 of the present invention recites a method, executed on a computing device, to perform an operation on extracted elements of a first software code, comprising the steps of "generating a list of desired elements of the first software code, the first code having a predefined command structure, *the predefined command structure being displayed via a graphical user interface*" and "extracting the desired elements from the first code; and performing an operation on the extracted elements."

Claim 1 incorporated the limitations of dependent claim 10. The Examiner rejected claim 10, stating that the limitations described therein are "considered inherent via the information cited in the rejections of claims 7-9." (See 6/15/04 Final Office Action, ¶ 2, page 5). The rejection of claim 7 cites the rejection of claim 2, which references use of a graphical use interface for receiving parameter and handler function information. In the rejection of claim 2, the Examiner equated the I/O (input/output) interface depicted in the Ravichandran reference with the graphical user interface ("GUI") recited in claim 1. However, a GUI as referenced in claim 1 is not equivalent to an I/O interface by definition, nor could it be considered equivalent with respect to Ravichandran's description.

A graphical user interface takes advantage of the computer's graphics capabilities to make a program easier to use. It may potentially free the user from learning and using complex command language, while allowing for the input and output of information to and from the essential computer. An I/O device is one that transfers information to/from a computer and to/from a peripheral device. Every transfer is an output from one device and an input into another. This differs significantly from a GUI, where data is input directly to the interface and the output is displayed on the same.

As described in the Ravichandran reference, the I/O interface 120 "facilitates communication between" a network interface 112, a target processor 114, a primary storage 116, and a secondary storage 118. (Ravichandran, col. 4, lines 22-25). As would be understood by one of ordinary skill in the art, this interfacing could not be performed solely by a GUI. Also, I/O interface 120 is not presented graphically to the user.

In light of the above distinctions, it is respectfully submitted that a GUI is not equivalent to an I/O (input/output) interface, as the Examiner suggests. As such, the "predefined command structure... displayed via a graphical user interface" cited in claim 1 is not anticipated by Ravichandran's disclosure of the I/O interface 120. Therefore, Appellants request that the rejection of claim 1 be reversed. Because claims 2-9 depend from, and therefore include all the limitations of claim 1, it is respectfully submitted that these claims are allowable for the same reasons as stated above.

The Examiner rejected claims 11-14 in view of the rejections of method claims 1-4.

(6/15/04 Office Action, ¶ 2, page 5). Claim 11 has been amended to include the limitations of claim 19. As amended, claim 11 recites limitations similar to those is amended claim 1. Such limitations include "a first engine receiving a list of desired elements of a first software code, the first code having a predefines command structure, *the predefined command structure being displayed via a graphical user interface.*" Therefore, Appellants respectfully submit that claim 11 is allowable for at least the reasons discussed above with regard to claim 1. Because claims 12-18 depend from, and therefore include all the limitations of claim 11, it is respectfully submitted that these claims are allowable for the same reasons as stated above.

9. Conclusions

For the reasons set forth above, Appellants respectfully request that the Board reverse the final rejections of the claims by the Examiner under 35 U.S.C. § 102(b) and indicate that claims 1-9 and 11-18 are allowable.

Respectfully submitted,

FAY, KAPLUN & MARCIN, L.L.P.

Dated: 1/12/05

By: [illegible]
Michael J. Marcin (Reg. No. 48,198)

FAY KAPLUN & MARCIN, LLP
150 Broadway, Suite 702
New York, NY 10038
(212) 619-6000 (phone)
(212) 619-0276 (facsimile)

APPENDIX A – APPEALED CLAIMS

1. A method executed on a computing device to perform an operation on extracted elements of a first software code, comprising the steps of:

generating a list of desired elements of the first software code, the first code having a predefined command structure, the predefined command structure being displayed via a graphical user interface;

extracting the desired elements from the first code; and

performing an operation on the extracted elements.

2. The method according to claim 1, wherein the code is generated according to the following substeps:

receiving parameter information via the graphical user interface,

receiving handler function information via the graphical user interface, and

automatically generating the first code using the parameter information and handler function information.

3. The method according to claim 1, wherein the list of desired elements includes a list of language translatable elements and wherein the performing step includes the following substeps:

translating the extracted elements from a first language into a second language.

4. The method according to claim 3, wherein the performing step includes the following substep:

inserting the translated elements back into the first code.

5. The method according to claim 3, wherein the performing step includes the following substep:

generating a second code as a function of the first code and the translated elements.

6. The method according to claim 1, wherein the list of desired elements includes a list of help-related elements and wherein the performing step includes the following substeps:

generating a help manual as a function of the extracted elements.

7. The method according to claim 1, wherein the list of desired elements is generated via the graphical user interface.

8. The method according to claim 1, wherein the graphical user interface displays the extracted elements.

9. The method according to claim 1, wherein the predefined command structure is a

hierarchical command tree.

11. A system, comprising:

a first engine receiving a list of desired elements of a first software code, the first code having a predefined command structure, the predefined command structure being displayed via a graphical user interface;

a second engine extracting the desired elements from the first code; and

a third engine performing an operation on the extracted elements.

12. The system according to claim 11, further comprising:

a code generation engine receiving parameter and handler function information via the graphical user interface and automatically generating the first code using the parameter and handler function information.

13. The system according to claim 11, wherein the list of desired elements includes a list of language translatable elements and wherein the third engine translates the extracted elements from a first language into a second language.

14. The system according to claim 13, wherein the third engine inserts the translated elements back into the first code to generate a second code.

15. The system according to claim 11, wherein the list of desired elements includes a list of help-related elements and wherein the third engine generates a documentation manual as a function of the extracted elements.

16. The system according to claim 11, wherein the list of desired elements is generated by the first engine via the graphical user interface.

17. The system according to claim 11, wherein the graphical user interface displays the extracted elements.

18. The system according to claim 11, wherein the predefined command structure is a hierarchical command tree.